

Something fundamental is brewing...

- Increasing prevalence of mobile work, ad hoc teams and computers conversing with computers
- Growing numbers of embedded and mobile information appliances
 - *PDA's, cell phones, CrossPad, InfoPen...*
 - *Over 4 billion embedded processors sold per year*
- Rich and growing pico-cellular wireless technologies
 - *Bluetooth, HomeRF, 802.11, IrDA...*
 - *Bluetooth to produce a 9x9mm radio on a chip*
- Emerging technologies for dynamic service discovery
 - *Jini, Universal Plug-and-Play, Service Location Protocol...*
- Increasing use of Next-Generation Software Languages and Tools
 - *Java, Tcl, DCOM, JavaScript, REBOL...*

**...leading to a concept that ITL calls
Pervasive Computing**

Pervasive Computing: The Key Defining Properties

- Ubiquitous
 - Low-Cost
 - Embedded
 - Distributed
 - Non-intrusive
 - Innumerable
- Interconnected
 - Wired Core
 - Wireless Edge
- Dynamic
 - Mobile
 - Self-configuring



ITL Pervasive Computing Portfolio

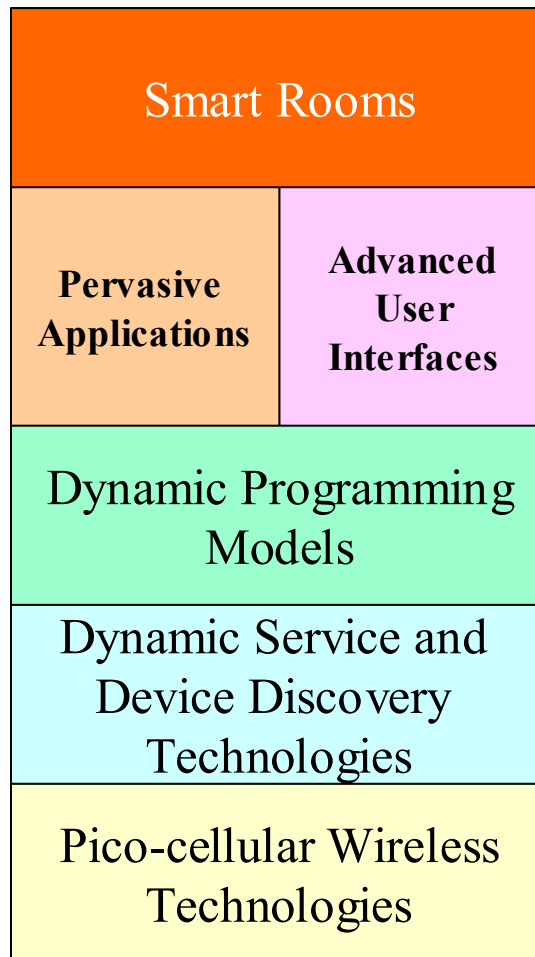
ITL Division*

Reference Model

User Interfaces
Division

Software
Division

Networking
Division



*These three divisions sponsored Pervasive Computing 2000, the first industry conference on this topic. And, of course, there is a large space for the Security Division

Focus of Networking Division



Relevant Industry Technologies

- Jini
- Service Location Protocol
- Universal Plug and Play
- Salutation Consortium
- Bluetooth Service Discovery

- IEEE 802.15 Wireless Personal Area Networks (WPAN)
- Bluetooth SIG
- HomeRF Consortium
- *Ultra Wideband Communications*

Pico-Cellular Wireless Technologies

- Network division has joined the Bluetooth SIG and the HomeRF Consortium
 - Reviewed, analyzed, and commented on the Bluetooth and HomeRF specs.
 - Develop an SDL model for the Bluetooth link layer
- Network division helped form IEEE 802.15 (WPAN) and helped to convince Bluetooth to submit its specification
 - Nada Golmie co-chairs the co-existence subgroup, which is working to understand co-interference among wireless technologies in the unlicensed 2.4 GHz band and to develop techniques for different technologies to co-exist within that band.
 - David Cypher submitted his Bluetooth model and continues to work with the industry players to improve the quality of the WPAN specifications
- Ultra-Wideband Wireless Communications Technology is expected to be permitted by the FCC soon
 - This technology will permit higher speeds over even smaller distances; thus, boosting the prospects for the application of mobile, pervasive computing technologies
 - Nader has been holding discussions with some parties that are interested in this technology

A Project in the ITL Pervasive Computing Portfolio

Assessing the state-of-the-art in Dynamic Discovery of Ad Hoc Network Services

Christopher Dabrowski, Olivier Mathieu, Kevin Mills, Doug Montgomery,
and Scott Rose

**ARDA Interim Review Meeting
December 21, 2000**

**Advanced Research and
Development Activity**

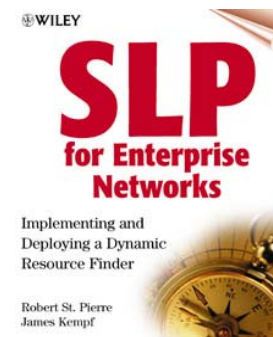
COTR is Greg Puffenbarger

Project Goal

Compare and contrast emerging commercial service discovery technologies with regard to function, structure, behavior, performance and scalability.



Plug and Play



Project Team

Modeling and Analysis

Christopher Dabrowksi, Architecture Description Languages and Tools

Kevin Mills, Scenarios, Metrics, and Properties

Measurement

Olivier Mathieu, Code Instrumentation

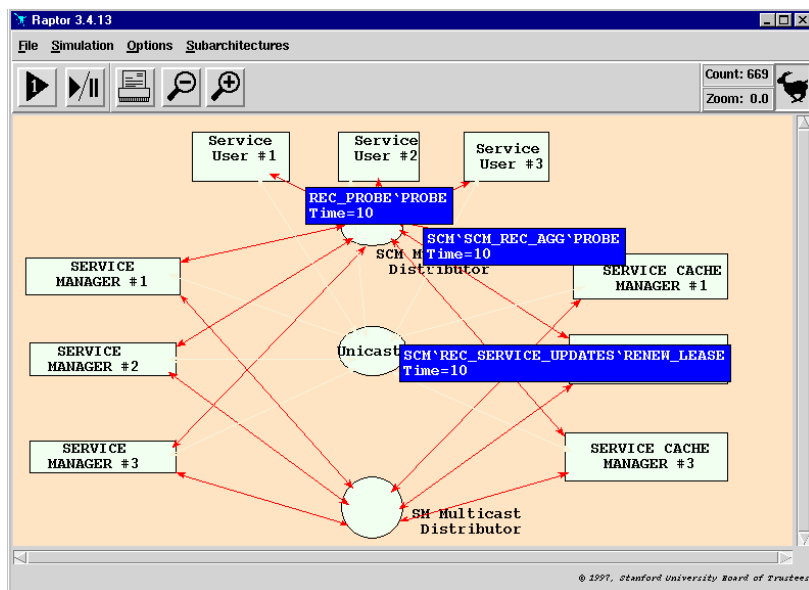
Doug Montgomery, Measurement Approaches and Techniques

Scott Rose, Generic Service Design and Implementation

Presentation Topics

- Planned Approach to Modeling and Analysis and Current Status
- Planned Approach to Measurement and Current Status
- Technical Discussion of Initial Progress
 - Generic and Specific UML Models Encompassing Jini, UPnP, SLP, HAVi, and Bluetooth (Saluation to be assessed later)
 - *Rapide* Model for Jini (90% complete)
 - Initial Measurement Testbed and Infrastructure Running for Jini and UPnP
- Upcoming Milestones and Planned Publications
- Demonstration

Modeling Function, Structure, and Behavior



Objectives

- (1) Provide increased understanding of the competing dynamic discovery services emerging in industry
- (2) Develop metrics for comparative analysis of different approaches to dynamic discovery and for analyzing consistency and completeness of discovery protocols
- (3) Assess suitability of architecture description languages to model and analyze emerging dynamic discovery protocols

Technical Approach

- Develop ADL models from selected specifications for service discovery protocols and develop a suite of scenarios and topologies with which to exercise the ADL models
- Propose a set of invariant properties that all dynamic discovery protocols should satisfy
- Propose a set of metrics, based on partially ordered sets, with which to compare and contrast discovery protocols
- Analyze the ADL models to search for flaws, to assess invariant satisfaction, and to compare and contrast protocols

Status as of December 21, 2000

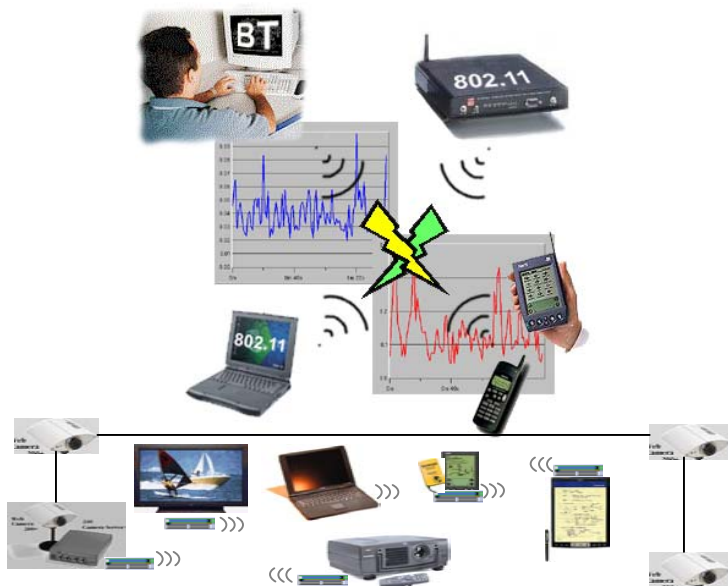
- Developed a generic UML model encompassing the structure and function of Jini, UPnP, SLP, Bluetooth, and HAVi
- Projected specific UML models for Jini, UPnP, and SLP
- Developed a Rapide Model of Jini structure, function, and behavior (90% complete)
- Drafted a scenario language to drive the Rapide Jini Model
- Developed some initial invariants and constraints for Jini behavioral model
- Discovered a number of ambiguities and inconsistencies in Jini Specification V1.1
- Discovered a major architectural flaw in the interaction between Jini directed discovery and multicast discovery

1/31/2002

Products

- Rapide specifications of Jini, Universal Plug and Play (UPnP), and Service Location Protocol (SLP)
- Scenarios and topologies for evaluating discovery protocols
- Suggested invariant properties for service discovery protocols
- Suggested metrics, based on partially ordered sets (POSETs), for comparing and contrasting discovery protocols
- Paper identifying flaws in Jini and UPnP and describing how these flaws were found
- Paper proposing invariants for service discovery protocols, and evaluating how Jini, UPnP, and SLP fare
- Paper comparing and contrasting Jini, UPnP, and SLP at the level of POSET metrics

Measuring Performance and Scalability



Objectives

- (1) Provide a quantitative, comparative analysis of the performance and scaling characteristics of emerging service discovery protocols (SDPs).
- (3) Design methodologies and tools for performance and scaling measurement of SDPs and supporting protocols.
- (4) Develop simulation tools for large scale ad-hoc network / application environments

Technical Approach

- Design and develop experimenters toolkits for conducting live performance analysis of SPDs implementations.
- Propose metrics and scenarios for comparing the performance of multiple SDP protocols.
- Design and develop simulation models of emerging SDPs and adhoc network environments.
- Analyze and compare the performance of SDPs based upon testbed measurements and simulation.

Status as of December 21, 2000

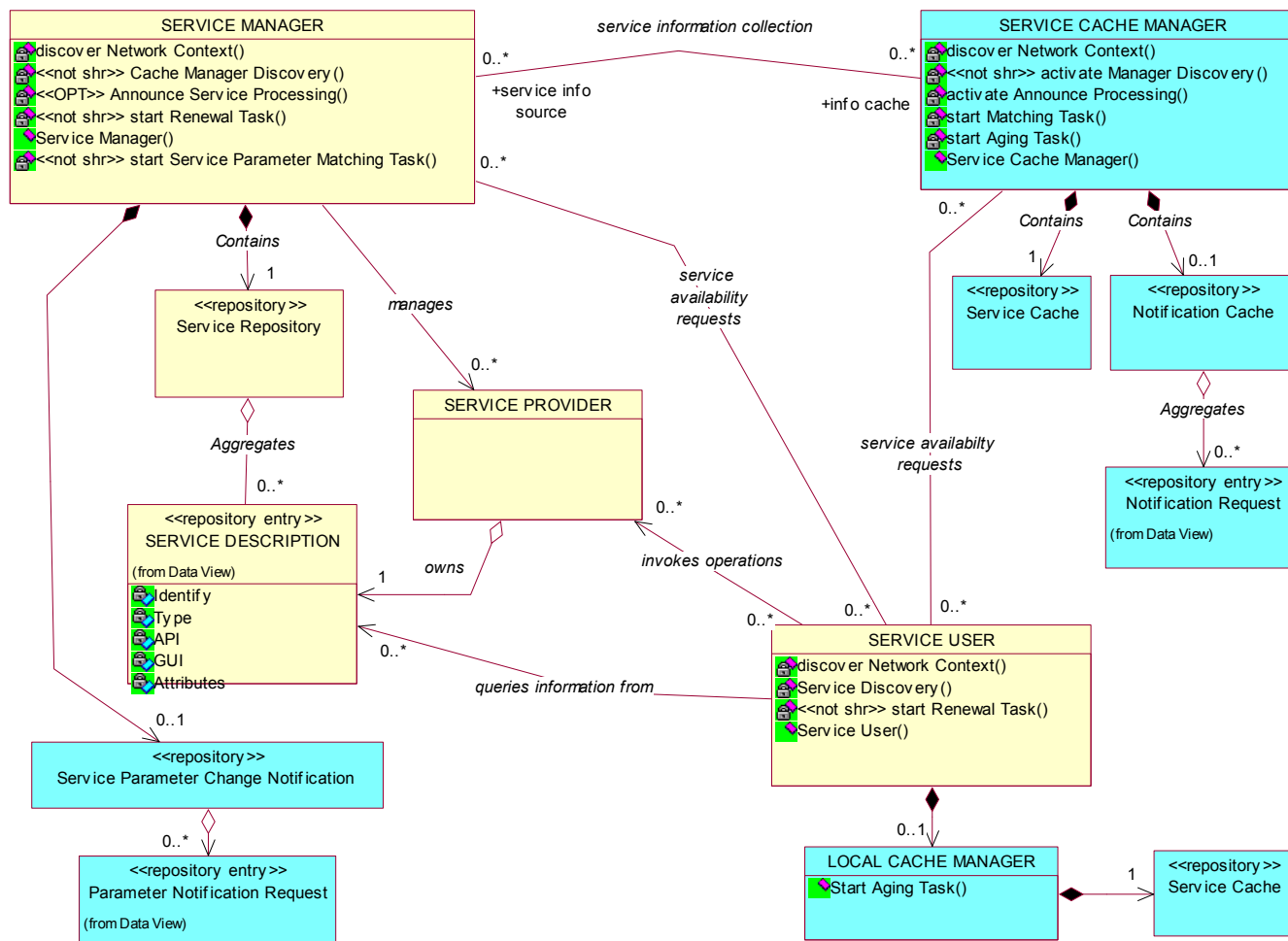
- Designed methodology and scenarios for comparative performance evaluation of live Jini and UPnP implementations.
- Established testbed with Sun Jini, Intel/Microsoft UPnP implementations.
- Developed synthetic workload generation tools for Jini and UPnP capable of emulating 10's-100's of devices/services and control point / clients.
- Discovered scaling problems with Intel Linux UPnP 1.0 implementation. Conducted initial investigations in protocol / parameter tuning to increase the scalability of this implementation.
- Began design and development of on-the-wire performance measurement tools for SDPs and supporting protocols.

1/31/2002

Products

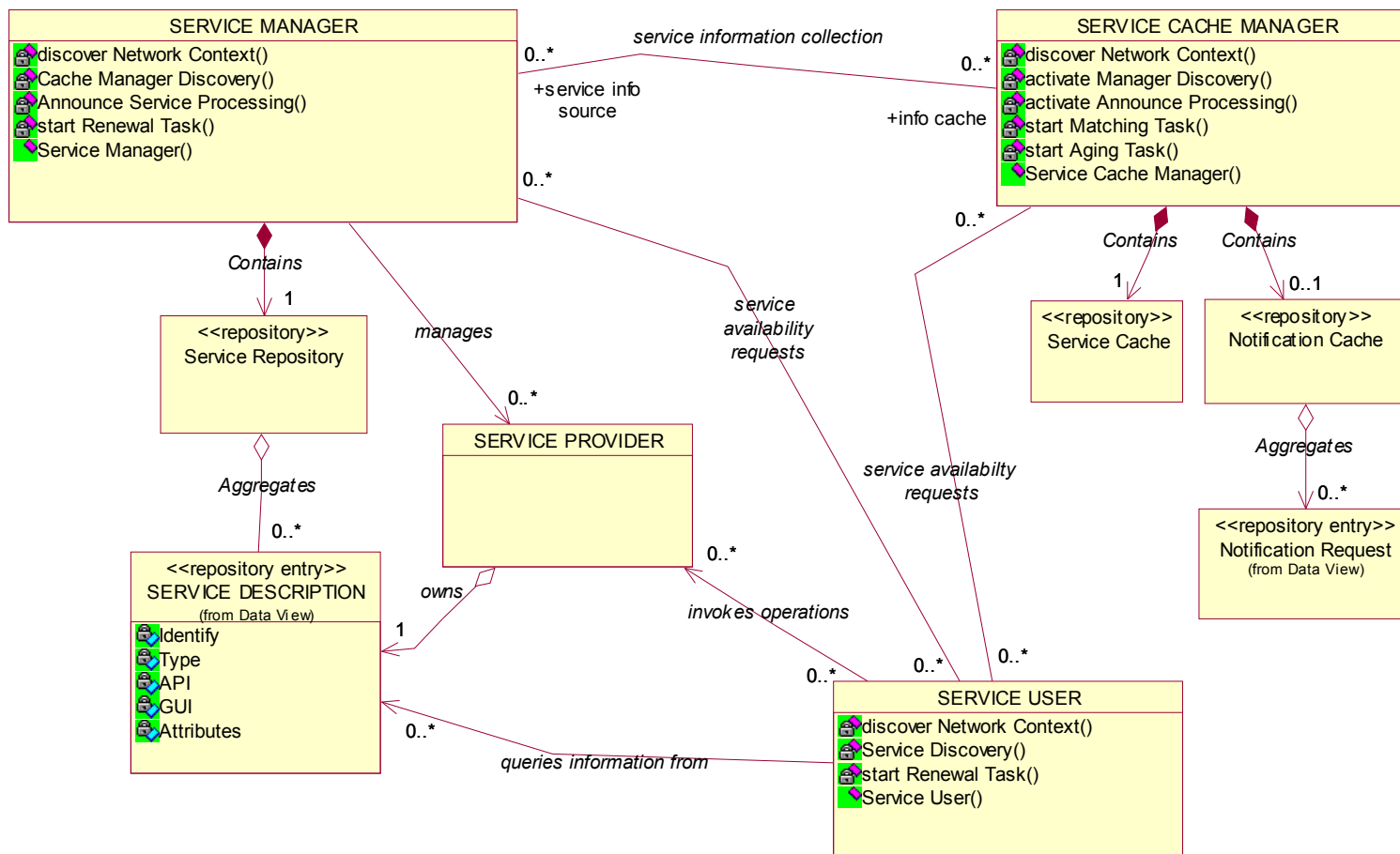
- Experimenter's toolkits consisting of synthetic workload generation tools, scenario scripts, and performance measurement tools for SDPs.
- Measurement methodologies and tools for SDPs and supporting protocols.
- Ad-hoc network simulation environment and SDP protocol models.
- Publications / standards contributions providing quantitative analysis of the relative performance and scaling properties of SDPs.

Generic UML Structural Model of Service Discovery Protocols

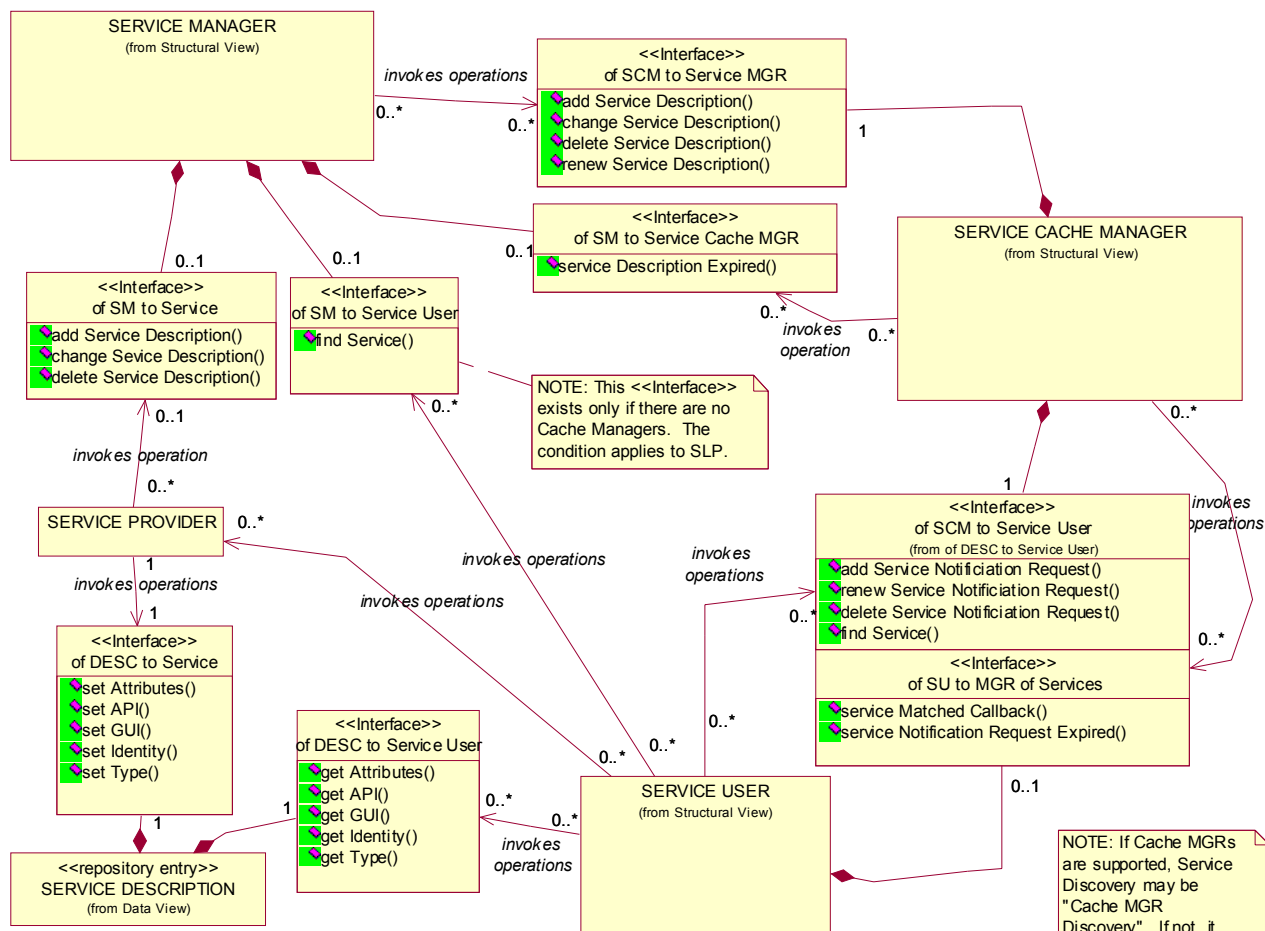


[illegible]

UML Structural Model of Jini



UML Functional Model of Jini



Architecture Description Languages and Tools

Allow us to **model the essential complexity** of discovery protocols, while ignoring the incidental complexity



Jini documented in a 385 page specification; however, the document is static and thus captures only the **normative complexity** because most of the essential complexity arises through interactions among distributed independently acting Jini components.



Incidental complexity represented by the code: for example consider Core Jini – an 832 page commentary on the massive amount of Java code that comprises Jini, which also depends on complex underlying code for Remote Method Invocation, Distributed Events, Object Serialization, TCP/IP, UDP, HTTP, and Multicast Protocol Implementation.

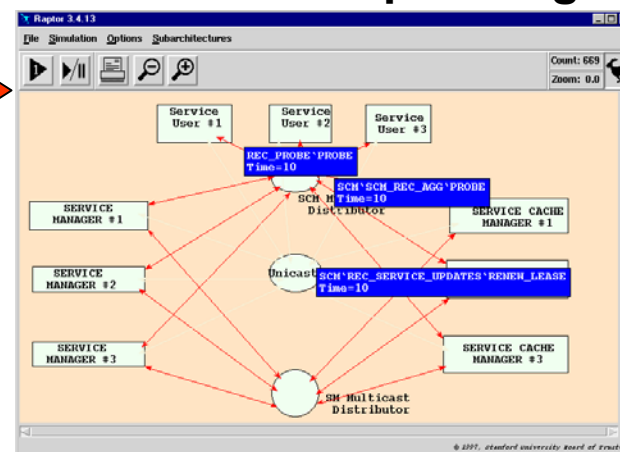
Rapide, an Architecture Description Language and Tools Developed for DARPA by Stanford

MODELING ESSENTIAL COMPLEXITY

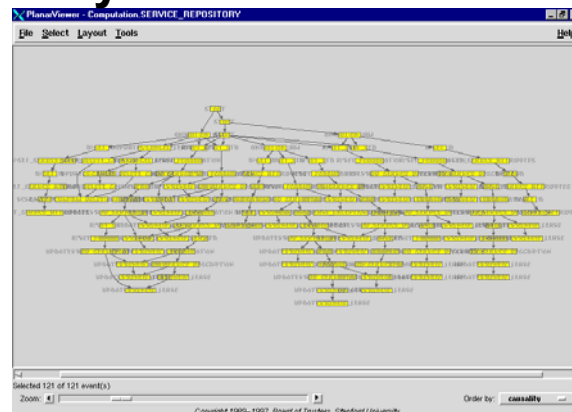
Model Specification in Rapide

```
-- *****
-- ** 3.3 DIRECTED DISCOVERY CLIENT INTERFACE **
-- *****
-- This is used by all JINI entities in directed
-- discovery mode. It is part of the SCM_Discovery
-- Module. Sends Unicast messages to SCMs on list of
-- SCMS to be discovered until all SCMS are found.
-- Receives updates from SCM DB of discovered SCMs and
-- removes SCMs accordingly
-- NOTE: Failure and recovery behavior are not
-- yet defined and need review.
TYPE Directed_Discovery_Client
(SourceID : IP_Address; InSCMsToDiscover : SCMList; StartOption : DD_Code;
 InRequestInterval : TimeUnit; InMaxNumTries : integer; InPV : ProtocolVersion)
IS INTERFACE
SERVICE DDC_SEND_DIR : DIRECTED_2_STEP_PROTOCOL;
SERVICE DISC_MODES : dual SCM_DISCOVERY_MODES;
SERVICE DD_SCM_Update : DD_SCM_Update;
SERVICE SCM_Update : SCM_Update;
SERVICE DB_Update : dual DB_Update;
SERVICE NODE_FAILURES : NODE_FAILURES; -- events for failure and recovery.
ACTION
IN Send_Requests(),
BeginDirectedDiscovery();
BEHAVIOR
action animation_lam (name: string);
MySourceID : VAR IP_Address;
PV : VAR ProtocolVersion;
```

Execute with Raptor Engine

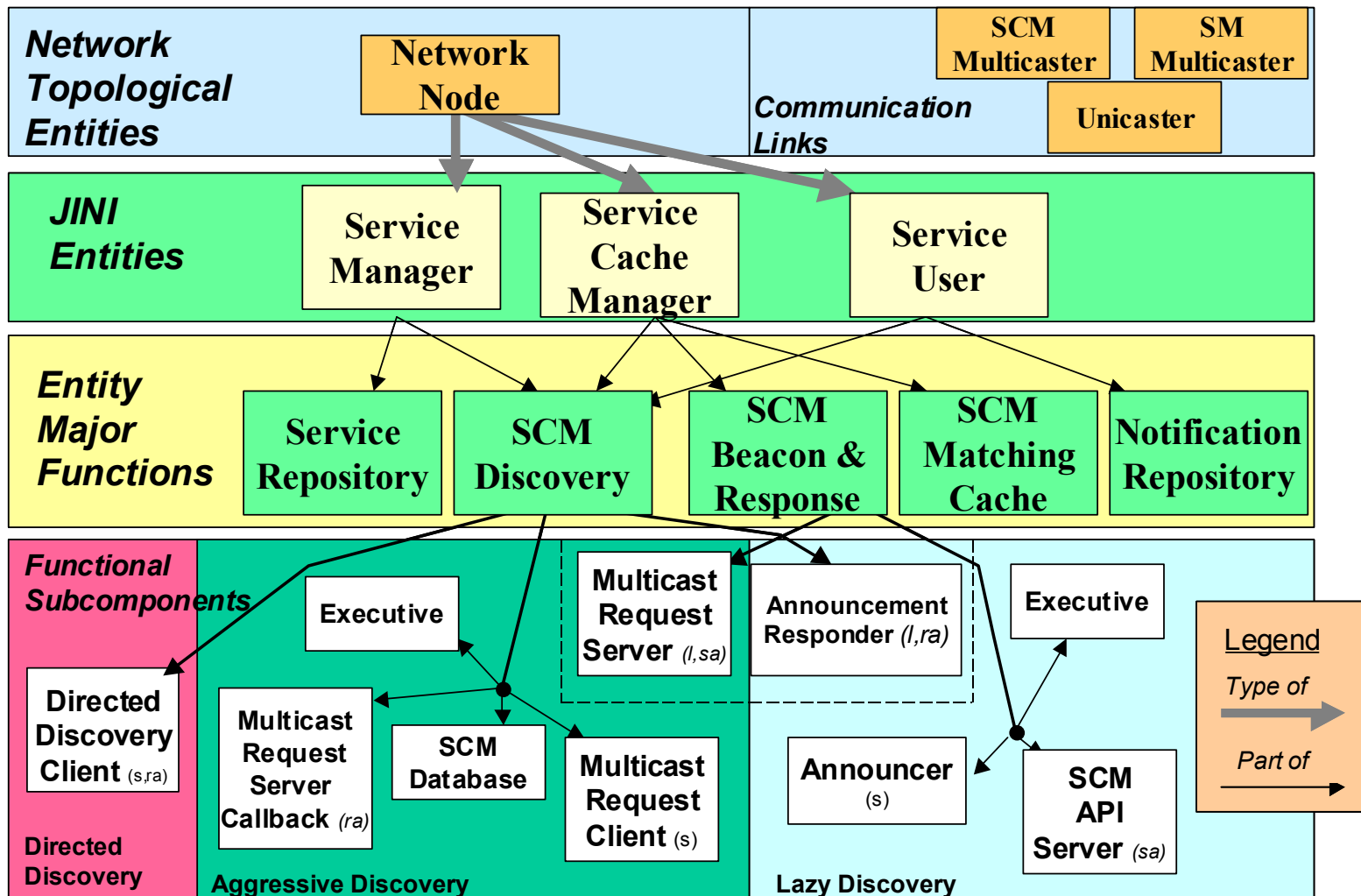


Analyze Generated POSETs

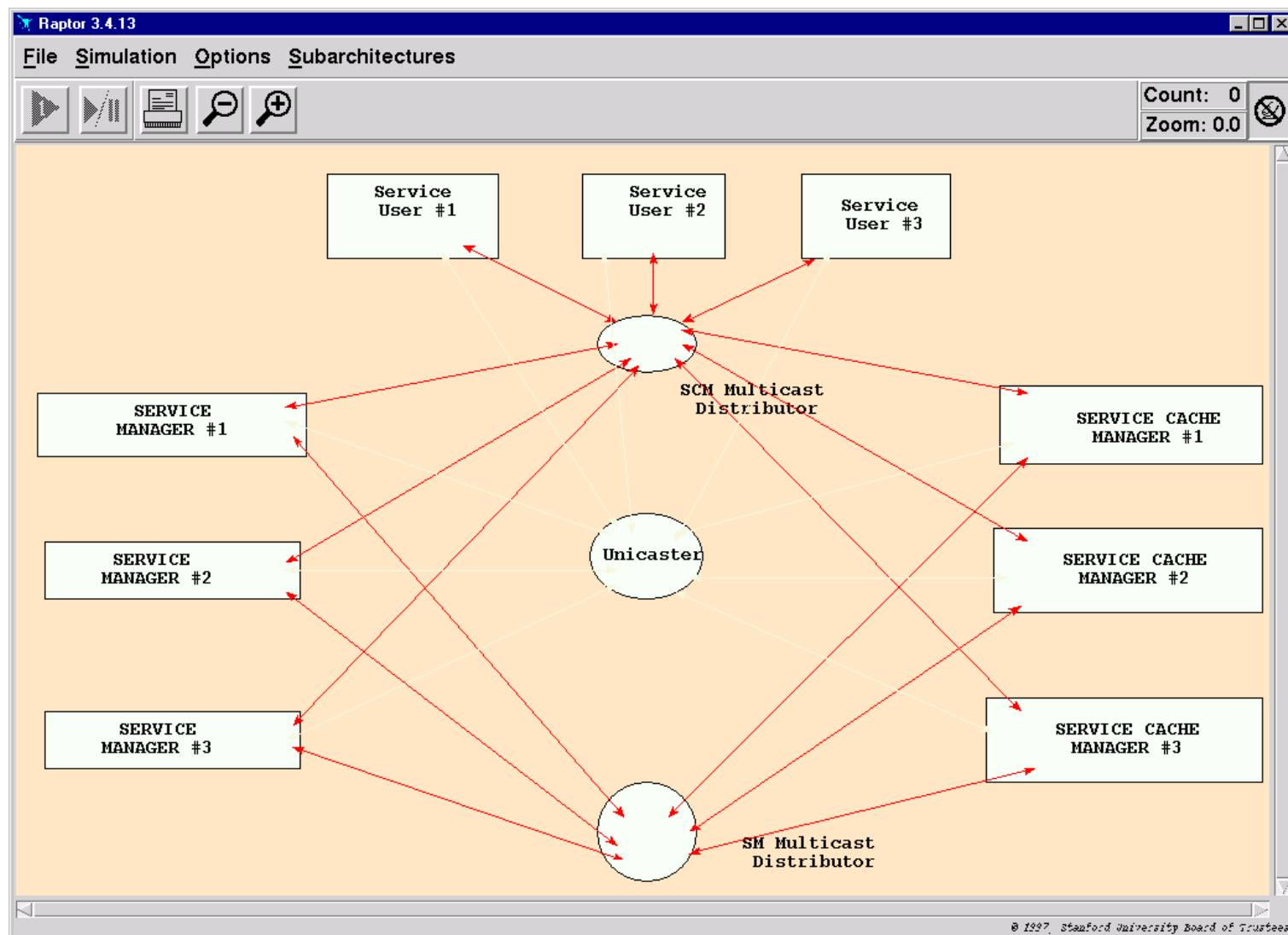


**Assess Invariant
Satisfaction &
Constraint
Violations**

Layered View of Prototype JINI Architecture in Rapide documented using SEI Architectural Layers Approach



Deploy a Model Topology with the Raptor Engine







Drive Model Topology with Scenarios

Node NodeID {on || off} [at time || after delay] [for duration]
Link LinkID {fail || restore} [at time || after delay] [for duration]
Mcast McastID {fail || restore} NodeID {receive || transmit} [at time ||
after delay] [for duration]
Group NodeID {leave || join} GroupID [at time || after delay] [for duration]
SCM NodeID {add || discard} SCMID [at time || after delay] [for duration]
Service NodeID add ServiceID [attributes] [leasetime] [at time || after delay]
[for duration]
Service NodeID {delete || expire} ServiceID [at time || after delay]
Service NodeID change ServiceID attributes [at time || after delay]
Find NodeID TemplateID [ServiceTemplate] on SCMID [MaxMatches]
[at time || after delay]
Notify NodeID add TemplateID [ServiceTemplate] [Transitions] [SCMID]
[leasetime] [at time || after delay] [for duration]
Notify NodeID {delete || expire} TemplateID [SCMID] [at time || after delay]

Analyze Invariant Satisfaction & Constraint Violations in Real-Time

Sample Invariants

⌚(SM  SD  SCM): (SM,SD) \models SCM registered-services
 \models SCM \models SM discovered-SCMs

⌚(SU  NR  SCM): (SU,NR) \models SCM registered-notifications
 \models SCM \models SU discovered-SCMs

SM is Service Manager

SD is Service Description

SCM is Service Cache Manager

SU is Service User

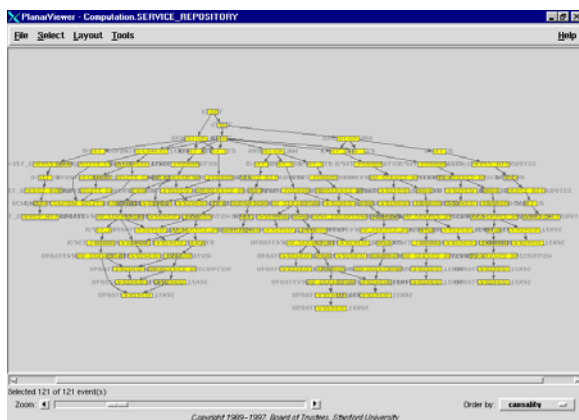
NR is Notification Request

registered-services is a set of (SM,SD) pairs

registered-notifications is a set of (SU,NR) pairs

discovered-SCMs is a set of SCM

Analyze POSETs Off-Line to Compare and Contrast Behaviors Given a Congruent Topology and Scenario



Metrics Based on Time

- Service latency?
- Service throughput?
- Recovery latency?

Metrics Based on Numbers of Messages

- Message volume?
- Message intensity?

Metrics Based on Complexity

- Degree of dependency among messages?
- Rate of constraint and invariant violations?
- Rate of exceptions?

Metrics Based on Change

- Derivative of the message intensity?
- Derivative of the service throughput?
- Derivative of the service latency?

SDP Performance / Scalability Measurements

Approach: Methodologies and tools for comparative performance and scaling analysis of live SDP implementations.

Initial focus - Jini and UPnP

- Design of technology independent **benchmark service**.
- Development of **synthetic workload generation tools** for emulating the behavior of large scale dynamic ad hoc networking environments.
- Development of implementation independent **performance measurement methodologies and tools** for SDPs and supporting protocols.

SDP Benchmark Service

- **Objective** – workload basis for meaningful comparative comparisons of Jini / UPnP performance.
 - Simple device / service that can be used to exercise all significant discovery / control capabilities of Jini and UPnP.
- **Benchmark Service** – very simple counting device.
 - Capabilities - Get / Set integer counter.
 - Attributes – GID, Name, Type
 - Enable multiple match / query semantics
 - Service interfaces
 - Control – get / set integer
 - GUI – simple user interface for control
 - Eventing – remote notification of counter change
- Jini and UPnP instantiations

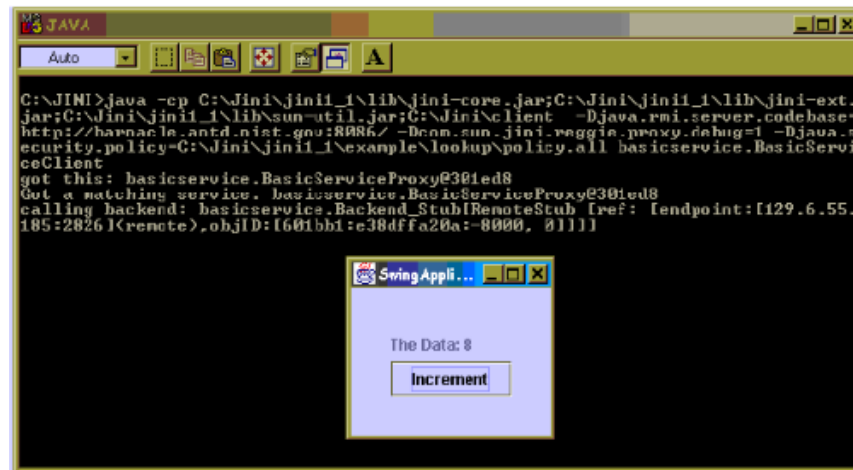
Jini Benchmark Service

```
/*
 * BasicService Interface
 * This is the interface for the Basic Jini service for
 * the client side.
 *
 * Scott Rose
 * NIST
 * 9/6/00
 */
```

```
package basicservice;
```

```
import java.rmi.RemoteException;
import net.jini.core.event.RemoteEventListener;
import net.jini.core.event.EventRegistration;
```

```
public interface BasicServiceIF
{
    public int getData() throws RemoteException;
    public void setData(int newVal) throws RemoteException;
    public EventRegistration addRemoteListener(RemoteEventListener rev)
        throws RemoteException;
    public void getGUI() throws RemoteException;
}
```



UPnP Benchmark Service

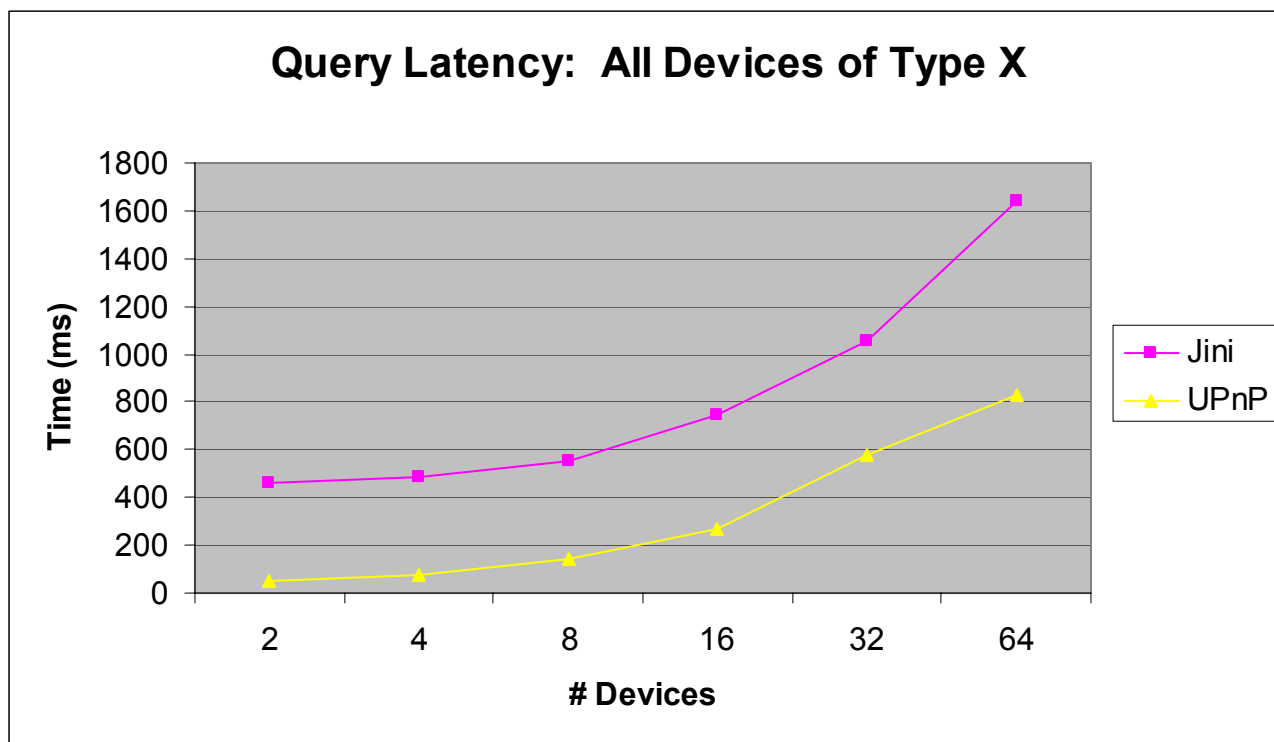
```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
<URLBase>http://129.6.51.81:20002</URLBase>
<device>
<deviceType>urn:schemas-upnp-org:device:basicdevice:1</deviceType>
  <friendlyName>Basic Service for Service Discovery Protocol
Testing</friendlyName>
  <manufacturer>NIST-ANTD-ITG</manufacturer>
  <manufacturerURL>http://w3.antd.nist.gov</manufacturerURL>
  <modelDescription>UPnP Basic Service 1.0</modelDescription>
  <modelName>BasicService</modelName> <modelName>1.0</modelName>
  <modelURL>http://w3.antd.nist.gov/modelURL</modelURL>
  <serialNumber>123456789001</serialNumber> <UDN>uuid:Upnp-BasicService-1_0-
darwin-20002</UDN>
  <UPC>123456789</UPC>
  <serviceList>
    <service>
      <serviceType>urn:schemas-upnp-
org:service:basicservice:1</serviceType>
      <serviceld>urn:upnp-org:serviceld:basicservice1</serviceld>
      <controlURL>/upnp/control/basicservice1</controlURL>
      <eventSubURL>/upnp/event/basicservice1</eventSubURL>
      <SCPDURL>/basicserviceSCPD.xml</SCPDURL>
    </service>
  </serviceList>
  <presentationURL>/basicdevice.html</presentationURL>
</device>
</root>
```

Synthetic Workload Generation Tools

- **Objective** – Emulate large, dynamic environments of 100's of devices / services and 10's of control points / clients.
 - Dynamic devices providing the benchmark service.
 - Scripted control points execute measurement scenarios.
- **Jini and UPnP Experimenters Toolkits**
 - Drive real implementations: SunMS Jini, Intel Linux & Windows ME UPnP.
 - Emulate the behavior of a large number of dynamic devices
 - # devices, device creation rate, device life time, service life time
 - Devices implement the benchmark service
 - Emulate the behavior control points / scripted behavior for testing
 - # clients, query workload – (query type, service names / types)
- **Jini / UPnP Device Emulation Tools**
 - Initial development complete – target of 100's of devices and 10's of control points met.
 - Discovered scaling problems in Intel Linux UPnP 1.0 SDK

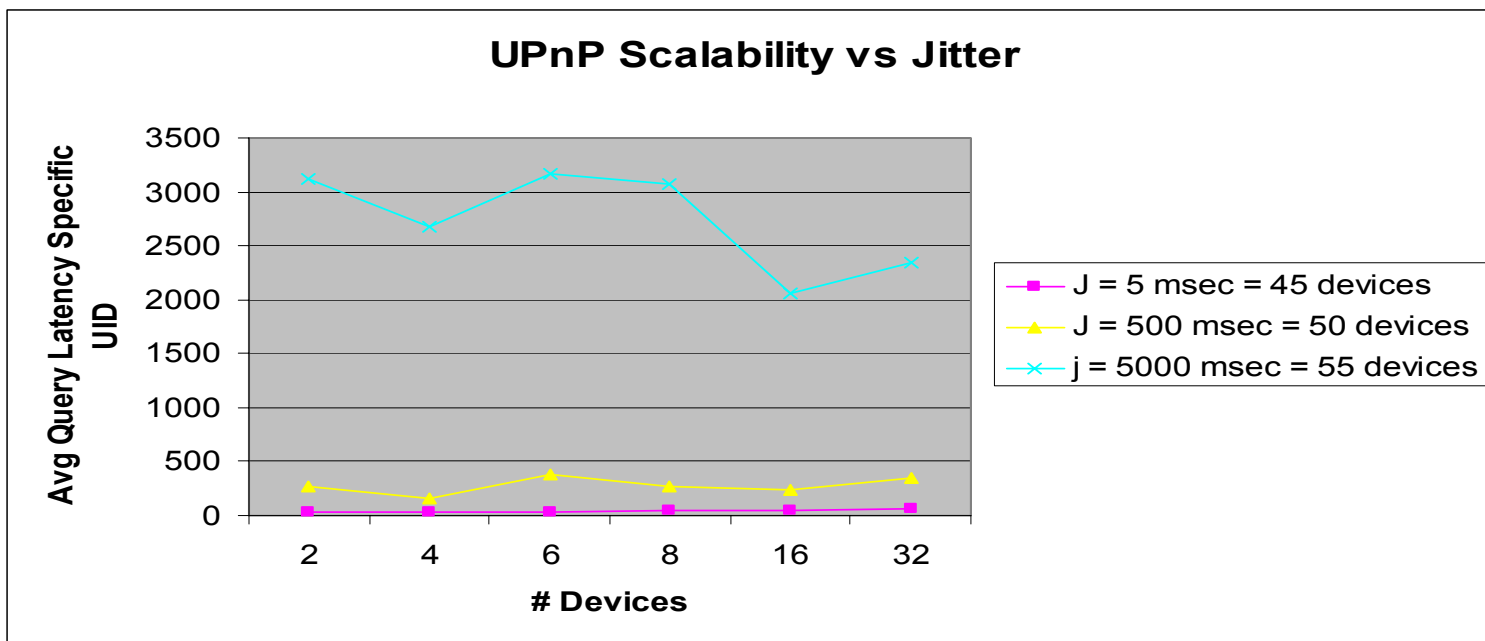
Some Example Results: Jini vs UPnP Discovery

- Example experiment: Query latency to find all devices of a given type.



Intel Linux UPnP Scaling Problems

- Problems encountered in achieving initial scaling goals for device emulation tools.
- UPnP scalability above 40 devices a function of protocol tuning parameters (e.g., response jitter, multicast retransmission factor).



Performance Measurement Methodologies

- Developed performance scenarios & metrics
 - Multiple service initiation
 - Client type query – single instance, multiple instances, all instances
 - Client instance query – query for existing service, persistent query
 - Client event notification – registration latency, notification latency
- Designing implementation independent on-the-wire performance (response/load) measurement tools.
 - How to measure HTTP/RMI based protocol transactions?

Modeling and Analysis: Upcoming Milestones and Publications

Milestones

- Jan 2001 – Complete Rapide Model for Jini, including scenario driver and specification of invariants and constraints
- Mar 2001 – Complete Rapide Model for Universal Plug-and-Play
- Jul 2001 – Complete Off-Line Analysis Tools for POSETs
- Aug 2001 – Complete Rapide Model for Service Location Protocol
- Oct 2001 – Partial analysis of Jini, UPnP, and SLP
- Dec 2001 – Complete analysis of Jini, UPnP, and SLP

Planned Papers

- Spring 2001 – Paper identifying flaws in Jini and UPnP and describing how those flaws were found
- Fall 2001 – Paper proposing invariants for service discovery protocols, and evaluating how Jini, UPnP, and SLP fare
- Winter 2002 – Paper comparing and contrasting Jini, UPnP, and SLP at the level of POSET metrics

Measurement: Upcoming Milestones and Contributions

Milestones

- Feb 2001 – Complete device / control point workload generation tools.
- Mar 2001 – Complete implementation independent measurement tools.
- May 2001 – Complete testbed performance measurement analysis.
- July 2001 – Complete development of simulation environment.
- Sep 2001 – Complete simulation analysis of SDPs.

Planned Contributions

- Summer 2001– Public domain release of Jini/UPnP experimenters toolkit consisting of workload generation tools, scenario scripts, and performance measurement tools for SDPs and supporting protocols.
- Fall 2001 – Public domain release of simulation environment for ad-hoc networks and protocol models for Jini/UPnP.
- Fall 2001 – Publication providing a quantitative performance/scaling comparison of Jini/UPnP technologies.

Plan to Assess Scalability

- Use Rapide Models as a Basis to Construct Simulation Models for Jini, UPnP and SLP, Possibly using JavaSim (from Ohio State University) or SSFnet (from Rutgers)
- Use Results from Measurement Portion of the Project to Parameterize the Simulation Models of the Discovery Protocols
- Design Experiments to Assess the Effect of Large Service and Device Populations on Network Traffic

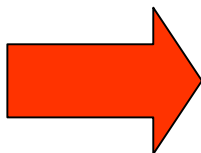
Modeling and Analysis Demonstration

Rapide Model of Jini V1.1

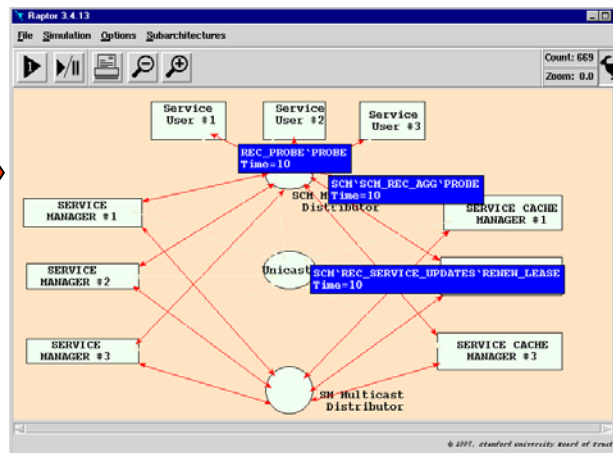
```

-- *****
-- ** 3.3 DIRECTED DISCOVERY CLIENT INTERFACE **
-- *****
-- This is used by all JINI entities in directed
-- discovery mode. It is part of the SCM_Discovery
-- Module. Sends Unicast messages to SCMs on list of
-- SCMS to be discovered until all SCMS are found.
-- Receives updates from SCM DB of discovered SCMs and
-- removes SCMs accordingly
-- NOTE: Failure and recovery behavior are not
-- yet defined and need review.
TYPE Directed_Discovery_Client
(SourceID : IP_Address; InSCMsToDiscover : SCMList; StartOption : DD_Code;
 InRequestInterval : TimeUnit; InMaxNumTries : integer; InPV : ProtocolVersion)
IS INTERFACE
SERVICE DDC_SEND_DIR : DIRECTED_2_STEP_PROTOCOL;
SERVICE DISC_MODES : dual SCM_DISCOVERY_MODES;
SERVICE DD_SCM_Update : DD_SCM_Update;
SERVICE SCM_Update : SCM_Update;
SERVICE DB_Update : dual DB_Update;
SERVICE NODE_FAILURES : NODE_FAILURES; -- events for failure and recovery.
ACTION
IN Send_Requests(),
    BeginDirectedDiscovery();
BEHAVIOR
action animation_Iam (name: string);
MySourceID : VAR IP_Address;
PV : VAR ProtocolVersion;

```



Execute with Raptor Engine



Generate POSETs

